Pengwyn Documentation

Release 1.0

Silica

October 03, 2016

Contents

Version 0.0.1A

Date 06/11/13

CHAPTER 1

Introduction

This guide explains how to start developing with the Pengwyn in a few steps.

In the following chapters you will learn step by step how to install the SDK and connect the Pengwyn on the development computer.

This SDK runs under Ubuntu 10.4 LTS and it's based on Sitara SDK.

After the host configuration you will see:

- how to compile the essential firmware: the bootloader and the kernel
- how to create a basic file system and save it on a sd-card.
- how use the development environment Code Composer Studio v5 for creating a basic project, compiling and debugging it

Basic knowledge of Linux system, specifically Ubuntu, is required.

1.1 Platforms

Sitara SDK supports different platforms.

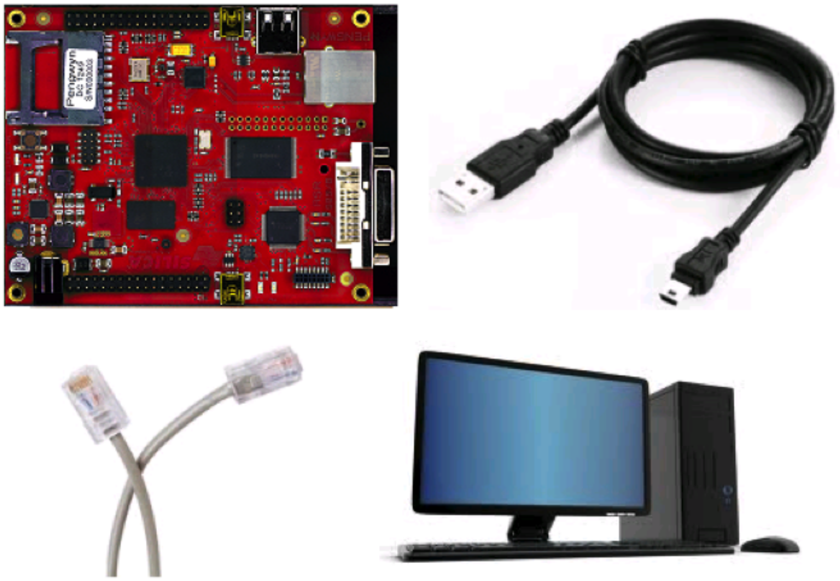
Platform	SDK	PSP	U-Boot	Kernel	Toolchain	Release Date
Pengwyn	5.06	04.06.00.09	2012.10	3.2	Gcc4.5.3	December 2012

You can get information about the other platforms at:kk

Platform	Document EVM	Provider
AM335xEVM	AM335xEVM	http://www.ti.com/
AM335x StarterKit (SK)	http://processors.wiki.ti.com/index.php/AM335xStarterKitHardwareUsersGuide	http://www.ti.com/tool/tmdssk3358

1.2 Hardware requirements

- Pengwyn board
- Ethernet cable
- Mini-USB type B cable
- Windows or Linux Host PC with at least 1GB (2GB recommended) of RAM and 40GB of free hard drive space

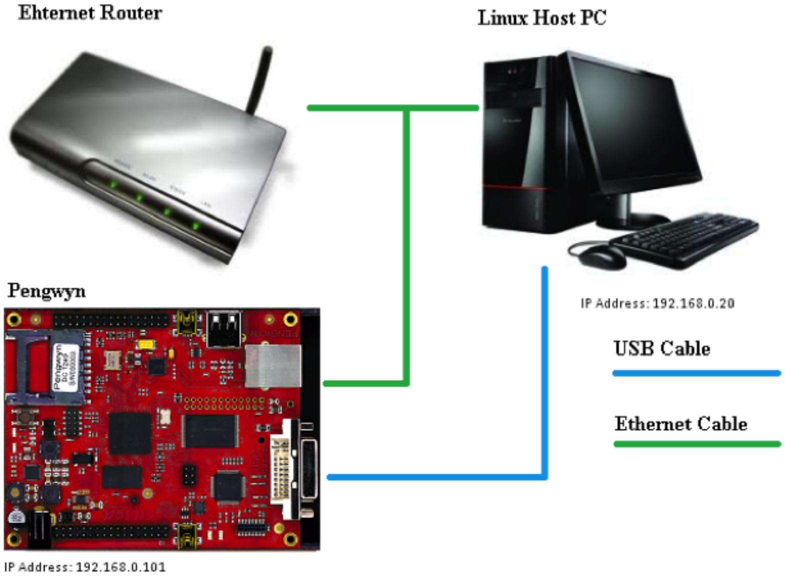


1.3 How to connect the board

Connect the USB cable to the host PC.

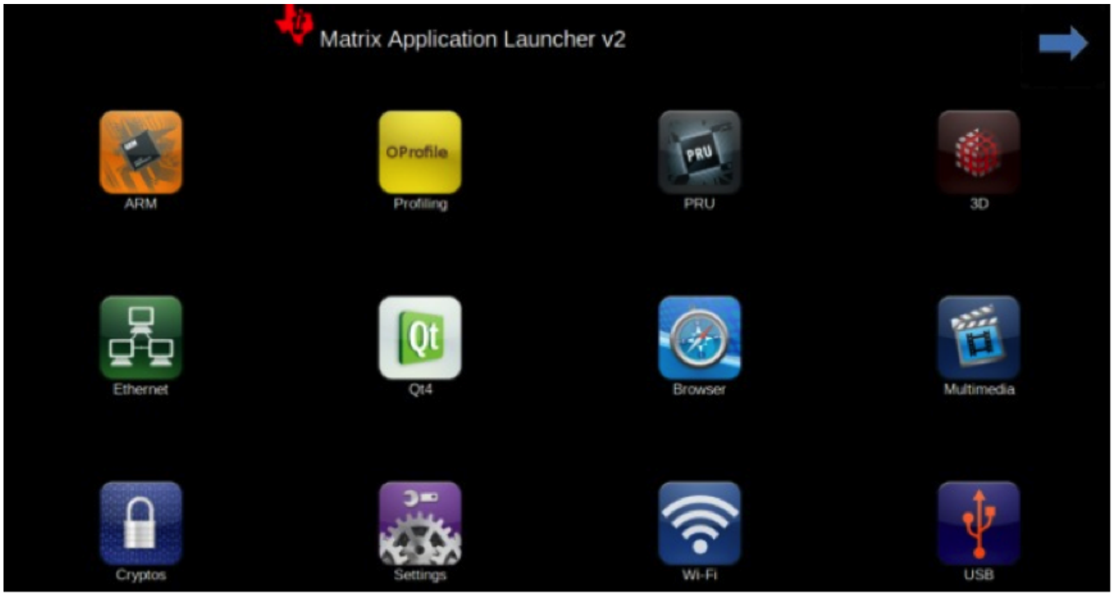
Create an Ethernet connection between Pengwyn and the board, in the way that these devices will be on the same LAN.

A possible configuration is showed below.



Linux System has been pre-flashed on the board.

If you turn on the Pengwyn and the display expansion board is connected the “Matrix Application Launcher v2” will appear.


Matrix Application Launcher v2
ARM
OProfile
Profiling
PRU
PRU
3D
Ethernet
Qt
Qt4
Browser
Multimedia
Cryptos
Settings
Wi-Fi
USB

CHAPTER 2

Pengwyn SDK

Contents:

2.1 Installing Virtual Machine

The development environment is provided as a Virtual Machine image.

To be able to start it you need first to install VirtualBox. Image has been created with VirtualBox version 4.2.6.

Go to:

https://www.virtualbox.org/wiki/Downloads

and download the version for Windows. You need also to download the Extension Pack.

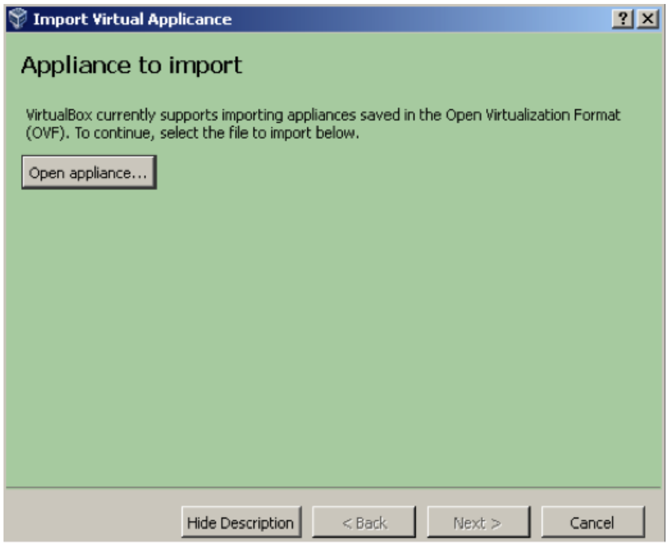
Important: Make sure that the extension pack has the same version of virtual box.

Install the software with all the default options.

Launch the program and follow the following steps:

Tip: If you are using Linux click directly on .ova file and skip to point 3.

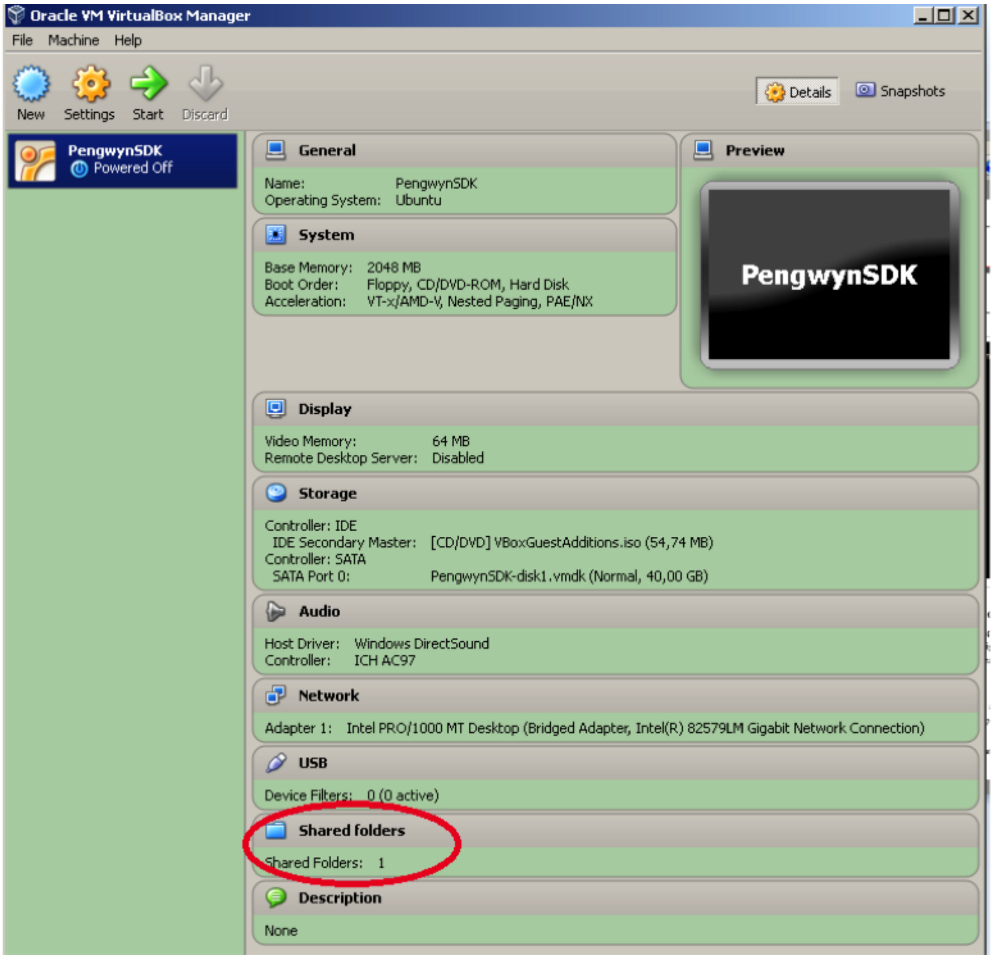
1. On the menu select: *File → Import Appliance*



Appliance to import

VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

2. Click on “Open appliance. . . ” button and select the iso file “PengwynSDK.ova”.
3. After opening the appliance, click on “Shared Folders” and select a folder to share with Windows.



General

Name: PengwynSDK
Operating System: Ubuntu

System

Base Memory: 2048 MB
Boot Order: Floppy, CD/DVD-ROM, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX

Display

Video Memory: 64 MB
Remote Desktop Server: Disabled

Storage

Controller: IDE
IDE Secondary Master: [CD/DVD] VBoxGuestAdditions.iso (54,74 MB)
Controller: SATA
SATA Port 0: PengwynSDK-disk1.vmdk (Normal, 40,00 GB)

Audio

Host Driver: Windows DirectSound
Controller: ICH AC97

Network

Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, Intel(R) 82579LM Gigabit Network Connection)

USB

Device Filters: 0 (0 active)

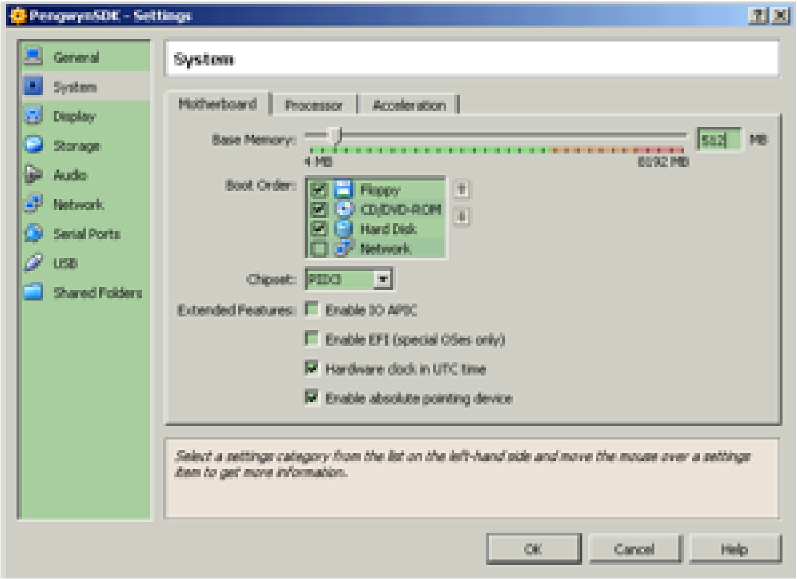
Shared folders

Shared Folders: 1

Description

None

4. If the host PC has only 1GB RAM go to “machine -> settings” menu and click on “System” tab and change RAM to 512MB.



System

Base Memory: 512 MB (4 MB – 8192 MB)

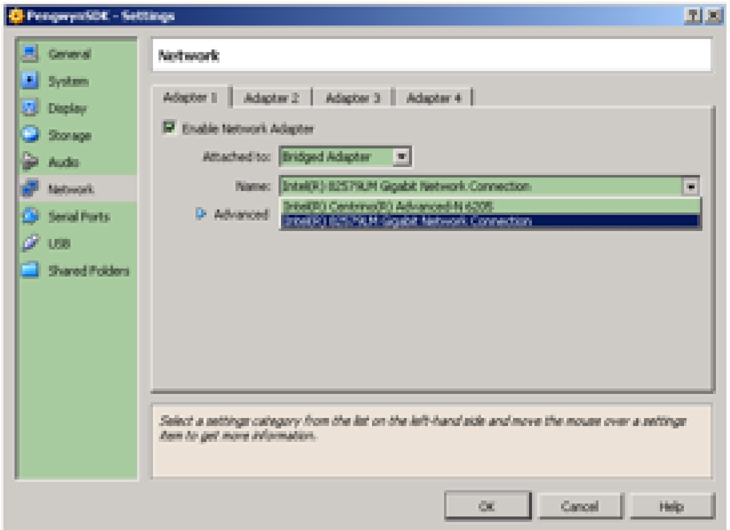
Boot Order:
☑ Floppy
☑ CD/DVD-ROM
☑ Hard Disk
☐ Network

Chipset: PIIX3

Extended Features:
☐ Enable IO APIC
☐ Enable EFI (special OSes only)
☑ Hardware clock in UTC time
☑ Enable absolute pointing device

Select a settings category from the list on the left-hand side and move the mouse over a settings item to get more information.

5. The ethernet card must be attached the LAN not the WLAN. To set the correct card, go to menu “machine -> Settings”. Click on “Network” tab and select your LAN card. Click Ok button to apply your choice.



Network

☑ Enable Network Adapter

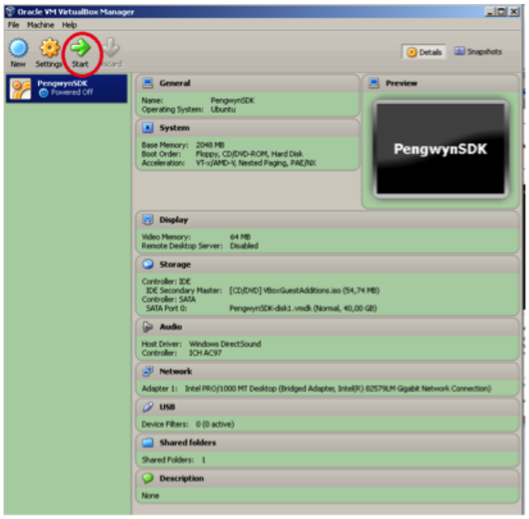
Attached to: Bridged Adapter

Name: Intel(R) 82579LM Gigabit Network Connection

Intel(R) Centrino(R) Advanced-N 6205
Intel(R) 82579LM Gigabit Network Connection

Select a settings category from the list on the left-hand side and move the mouse over a settings item to get more information.

6. Click the icon “Start” button on the toolbar.



General
Name: PengwynSDK
Operating System: Ubuntu

System
Base Memory: 2048 MB
Boot Order: Floppy, CD/DVD-ROM, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX

Display
Video Memory: 64 MB
Remote Desktop Server: Disabled

Storage
Controller: IDE
IDE Secondary Master: [CD/DVD] VBoxGuestAdditions.iso (54,74 MB)
Controller: SATA
SATA Port 0: PengwynSDK-disk1.vmdk (Normal, 40,00 GB)

Audio
Host Driver: Windows DirectSound
Controller: ICH AC97

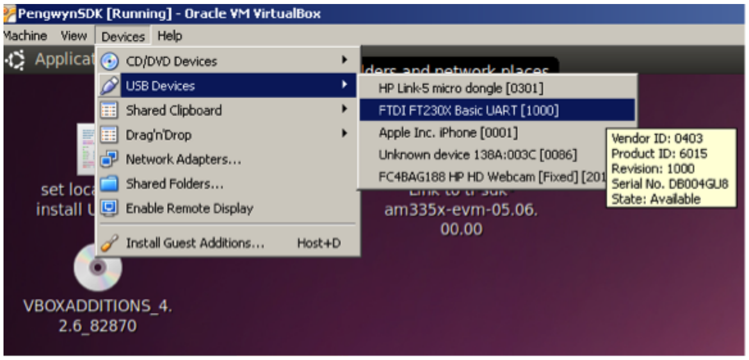
Network
Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, Intel(R) 82579LM Gigabit Network Connection)

USB
Device Filters: 0 (0 active)

Shared folders
Shared Folders: 1

Description
None

7. Every time you connect the mini-USB from the PC to the card. On menu click *Device* → *USB Devices* → *FTDI*.



Vendor ID: 0403
Product ID: 6015
Revision: 1000
Serial No. DB004GU8
State: Available

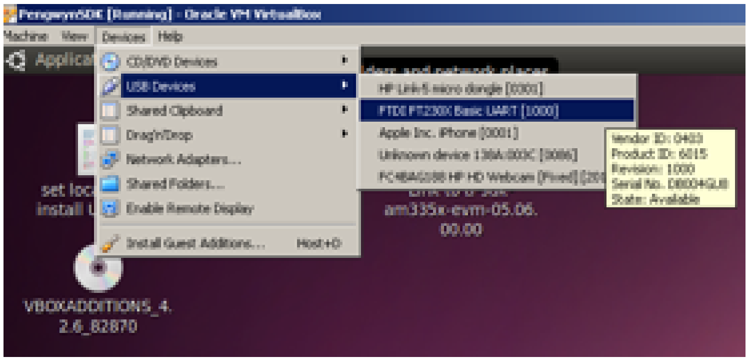
8. The default keyboard is set to USA layout.

If your keyboard layout is different, to change it, from the menu of Ubuntu:

go to “System -> preferences -> keyboard”.

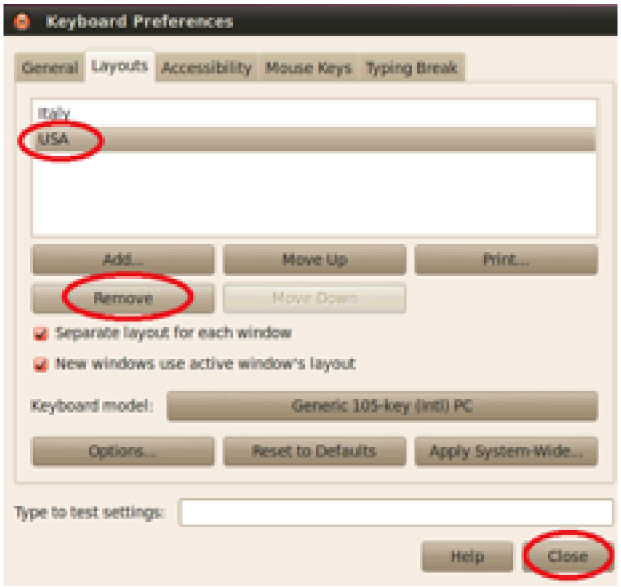
Select “Layout” tab.

Click on “Add...” button.



Vendor ID: 0403
Product ID: 6015
Revision: 1000
Serial No. DB004GU8
State: Available

Select your keyboard layout and press “add” button. Then select “USA” and click “Remove Button”.



Press “Close” button.

9. Open a terminal (ctrl + alt + t) and type the following command:

```
sudo usermod -a -G vboxsf pengwyn
```

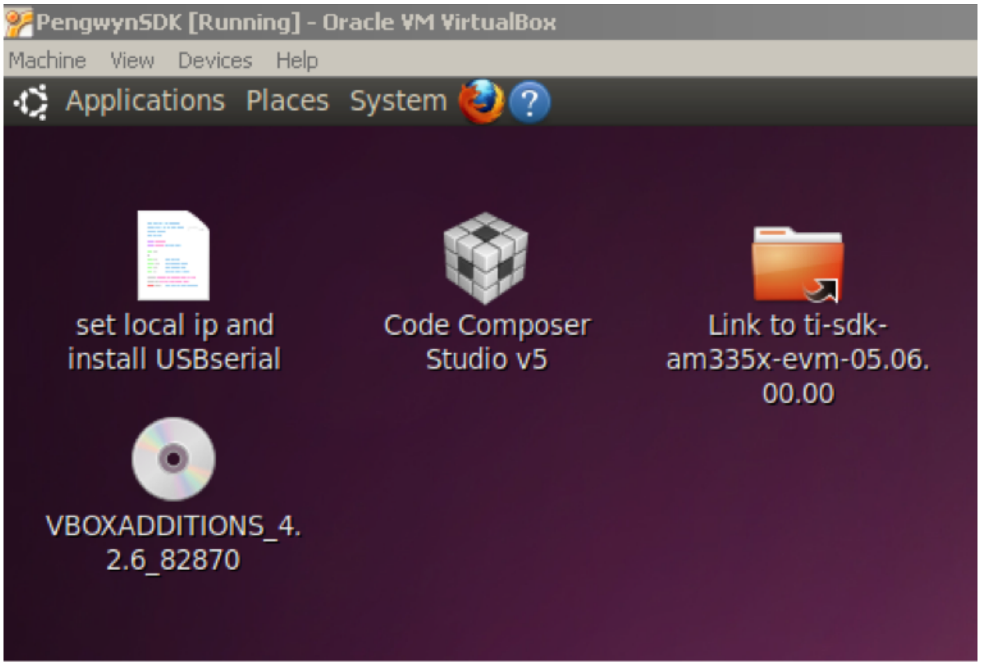
User name is **pengwyn**, password is **pengwyn**.

Ubuntu will be launched with the Sitara SDK integrated ready to be used. No installation or other configuration is required.

2.2 SDK Structure Overview

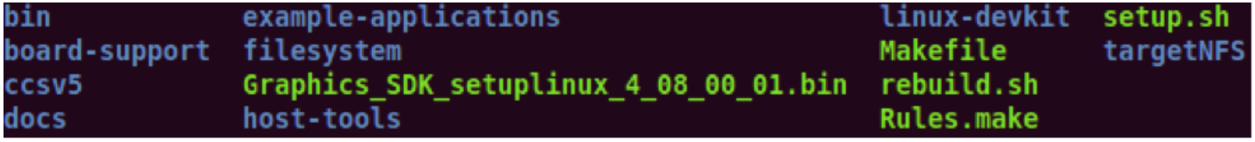
At the start of the Ubuntu desktop, there are three icons:

- “set local ip and install USBserial”: sets the local IP 192.168.0.20 and enables serial connection to the board
- Code Composer Studio v5: Eclipse based IDE used for both application & debug using gdbserver
- Link to ti-sdk-am335x-evm-05.06.00.00: link to the main directory that contains the Sitara Linux SDK


PengwynSDK [Running] - Oracle VM VirtualBox
Machine View Devices Help
Applications Places System
set local ip and install USBserial
Code Composer Studio v5
Link to ti-sdk-am335x-evm-05.06.00.00
VBOXADDITIONS_4.2.6_82870

The Sitara SDK directory contains the code and tools used to develop for Sitara devices.

You will find the following folders:



```
bin             example-applications                  linux-devkit  setup.sh
board-support   filesystem                            Makefile      targetNFS
ccsv5           Graphics_SDK_setuplinux_4_08_00_01.bin  rebuild.sh
docs            host-tools                            Rules.make
```

bin: Contains the helper scripts for configuring the host system and target device. Most of these scripts are used by the setup.sh script.

board-support: Contains the SDK components that need to be modified when porting to a custom platform. This includes the kernel and boot loaders as well as any out of tree drivers.

docs: Contains various SDK documentation such as the software manifest and additional user's guide. This is also the location where you can find the training directory with the device training material.

Example-applications: Contains the sources of the TI example applications, as seen during the out-of-box demonstration.

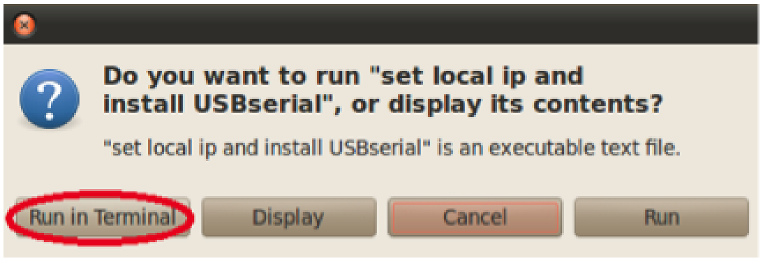
filesystem: Contains the reference file systems. These include the smaller base file system as well as the full-featured SDK file system.

host-tools: Contains the host side tools such as pinmux and flash tool.

linux-devkit: Contains the cross-compile toolchain and libraries to speed up the development for the target device. Graphics_SDK_setuplinux_<version>.bin: This is the installer for the graphics SDK. The graphics SDK components are used by the Sitara Linux SDK to provide additional demos and pre-built Qt libraries to accelerate various Qt functions.

2.3 Connect host PC to Pengwyn board

For connecting the board click the icon "set local ip and install USBserial" and push "Run in Terminal" button. Enter the password: pengwyn.



Do you want to run "set local ip and install USBserial", or display its contents?

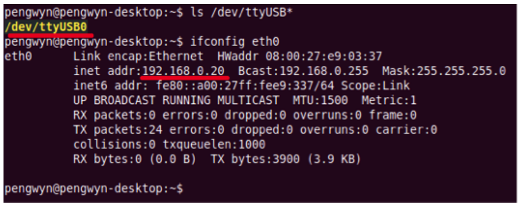
"set local ip and install USBserial" is an executable text file.

This script will set the IP address of the host pc to 192.168.0.20 and it will enable USB connection.

To check the configuration is correct use these commands:

```
ls /dev/ttyUSB*
ifconfig eth0
```

If all is ok you will see:

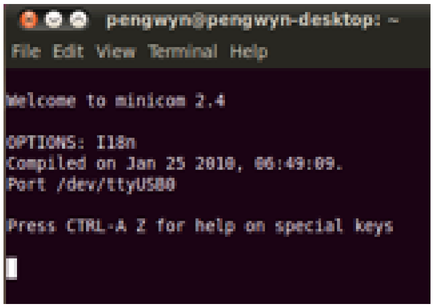


```
pengwyn@pengwyn-desktop:~$ ls /dev/ttyUSB*
/dev/ttyUSB0
pengwyn@pengwyn-desktop:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 08:00:27:e9:03:37
          inet addr:192.168.0.20  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fee9:337/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:3900 (3.9 KB)

pengwyn@pengwyn-desktop:~$
```

Open a terminal (ctrl + alt + t) and open Minicom with the following options:

```
minicom -w -s
```

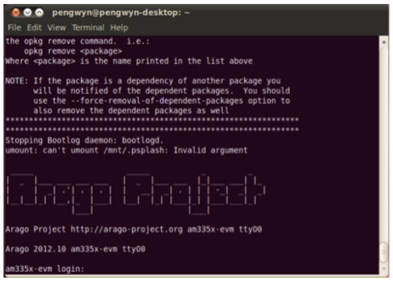
If all works correctly you will se this screen:



```
Welcome to minicom 2.4

OPTIONS: I18n
Compiled on Jan 25 2010, 06:49:09.
Port /dev/ttyUSB0

Press CTRL-A Z for help on special keys
```

Reset the board with the reset button on the board, located near the sd-card slot. You will see the startup of the Pengwyn on the terminal.

At login insert: root and press enter.



```
the opkg remove command.  i.e.:
    opkg remove <package>
Where <package> is the name printed in the list above

NOTE: If the package is a dependency of another package you
      will be notified of the dependent packages.  You should
      use the --force-removal-of-dependent-packages option to
      also remove the dependent packages as well
************************************************************
************************************************************
Stopping Bootlog daemon: bootlogd.
umount: can't umount /mnt/.psplash: Invalid argument

Arago Project http://arago-project.org am335x-evm ttyO0

Arago 2012.10 am335x-evm ttyO0

am335x-evm login:
```

Configure the IP address with the command:

```
ifconfig eth0 192.168.0.101
```

Now the connection is completed.

2.4 Compile U-Boot, MLO and Kernel

This document covers the general use of U-Boot v2012.10 and the AMSDK on the following platforms:

am335x EVM am335x EVM-SK Pengwyn

Before starting to build the projects open a terminal (ctrl + alt + t) and go to the root folder of the SDK Sitara:

```
cd ti-sdk-am335x-evm-05.06.00.00
```

Type "cd ti-", and press the "tab" key for autocompletion. Create a folder to copy once completed system images.:

```
cd board-support
```

In this folder you will find the projects that we're going to compile.:

```
mkdir built-images
```

This folder will be used as destination for the built images.

2.4.1 Build U-Boot & MLO

Now go into the project U-Boot:

```
cd u-boot-2012.10-psp05.06.00.00
```

As TI is suggesting, we, as well, recommend keeping the object files separated when building. You can do this using the following option parameter when invoking the make command:

```
O = object-directory
```

Where "object-directory" is the name of a folder you created for this purpose with:

```
mkdir object-directory
```

You can now compile the bootloader with the following commands:

```
rm -rf ./object-directory
export PATH="/home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/linux-devkit/bin:$PATH"
make O=object-directory CROSS_COMPILE=arm-arago-linux-gnueabi- ARCH=arm pengwyn
```

When the compilation ends you will find the bootloader files in the directory of the object files:

- u-boot.img
- MLO

Copy these files to built-images folder:

```
cp MLO ../../built-images
cp u-boot.img ../../built-images
```

2.4.2 Build the Kernel

From ti-sdk-am335x-evm-05.06.00.00 directory, go to the folder for the development of the operating system:

```
cd board-support
```

In this folder there are the projects t we're going to compile. Now go into the Kernel:

```
cd linux-3.2.0-psp05.06.00.08
export PATH="/home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/linux-devkit/bin:$PATH"
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- uImage
```

When compilation finish do:

```
cd arch/arm/boot
```

There you will find the compiled kernel "uImage", copy it to the folder you created previously (built-images):

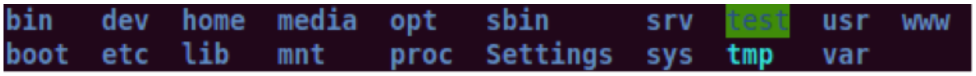
```
cp uImage /home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/board-support/built-images
```

You do not need to copy the driver modules to the file system because the compiled kernel already includes them.

2.5 Filesystem

The operating system will use the filesystem in the directory:

/home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/targetNFS



```
bin   dev  home  media  opt   sbin      srv  test  usr  www
boot  etc  lib   mnt    proc  Settings  sys  tmp   var
```

Copy to this folder all the files you want in the distribution. In this tutorial we are not going to do any modification to the filesystem.

The only change that was made was adding the logo showed during the startup of the board.

2.6 Copy the Operating System to the SD

The purpose of this section is to load the operating system on the board. There are several possible solutions to boot Linux, for this tutorial we use a SD-CARD.

You need to copy to the SD-CARD: the bootloader (u-boot and MLO), uImage kernel and filesystem.

The first step is to be sure that the SD-CARD device is accessible from VirtualBox.

If your PC has a built-in slot for the SD-CARD, the VirtualBox probably will not detect it.

The solution is using an USB SD-CARD reader.

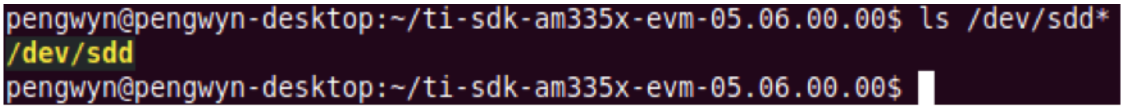
Connect the reader to your computer.

In the VirtualBox menu select Devices → USB Devices → "your adapter".

Insert the card into the adapter.

If everything works Ubuntu will recognize the card and will appear in "/dev" folder the device sdd.:

```
ls /dev/sdd*
```



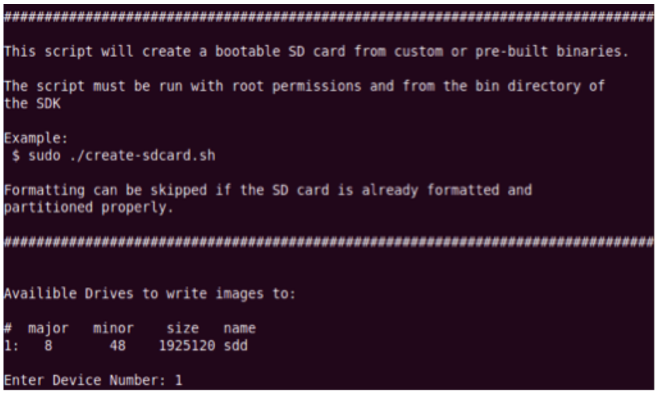
```
pengwyn@pengwyn-desktop:~/ti-sdk-am335x-evm-05.06.00.00$ ls /dev/sdd*
/dev/sdd
pengwyn@pengwyn-desktop:~/ti-sdk-am335x-evm-05.06.00.00$
```

In ti-sdk-am335x-evm-05.06.00.00 directory in bin folder there is script file, create-sdcard.sh: launch it.:

```
cd /home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/bin
sudo ./create-sdcard.sh
```

The password for sudo is: pengwyn.

If everything goes well you will the drive appears as available and ready to write the image. Select The drive:



```
################################################################################

This script will create a bootable SD card from custom or pre-built binaries.

The script must be run with root permissions and from the bin directory of
the SDK

Example:
 $ sudo ./create-sdcard.sh

Formatting can be skipped if the SD card is already formatted and
partitioned properly.

################################################################################


Availible Drives to write images to:

#  major   minor    size   name
1:   8       48    1925120 sdd

Enter Device Number: 1
```

Unmount the drive as required by the script:

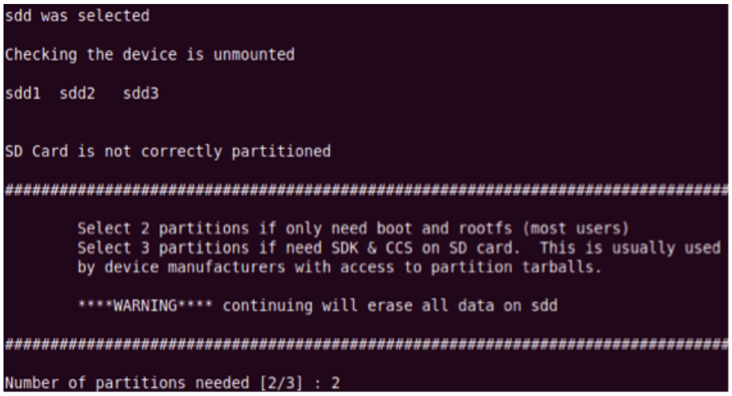
```
umount /dev/sddx
```

Where x is the number of device used.

Next the screen for partitioning the sd-card will appear. Select ‘y’.

Warning: Your sd-card will be formatted.**

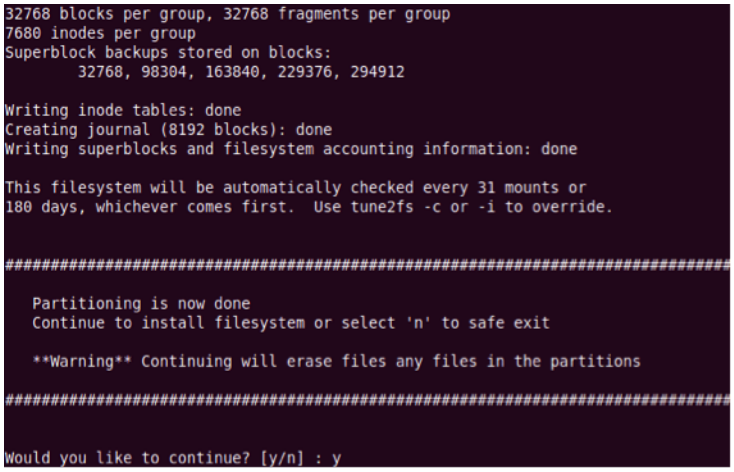
Select 2 partitions and wait until the partitioning process ends.



```
sdd was selected

Checking the device is unmounted

sdd1  sdd2   sdd3


SD Card is not correctly partitioned

################################################################################

        Select 2 partitions if only need boot and rootfs (most users)
        Select 3 partitions if need SDK & CCS on SD card.  This is usually used
        by device manufacturers with access to partition tarballs.

        ****WARNING**** continuing will erase all data on sdd

################################################################################

Number of partitions needed [2/3] : 2
```

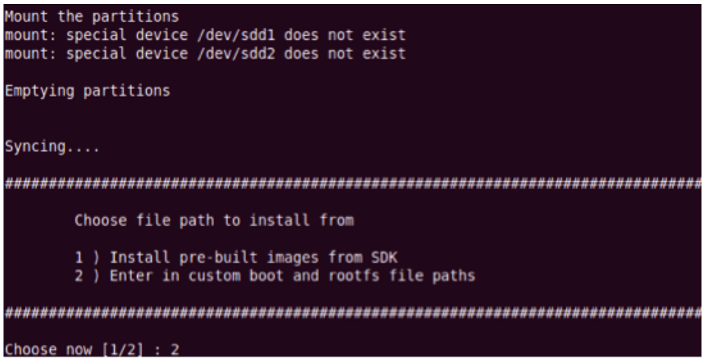
Now you need choose the path where the image files are located.

Select 2 for custom boot and rootfs paths.



```
32768 blocks per group, 32768 fragments per group
7680 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 31 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.


################################################################################

   Partitioning is now done
   Continue to install filesystem or select 'n' to safe exit

   **Warning** Continuing will erase files any files in the partitions

################################################################################


Would you like to continue? [y/n] : y
```

Press y key to continue. Type the path of the built-images folder.



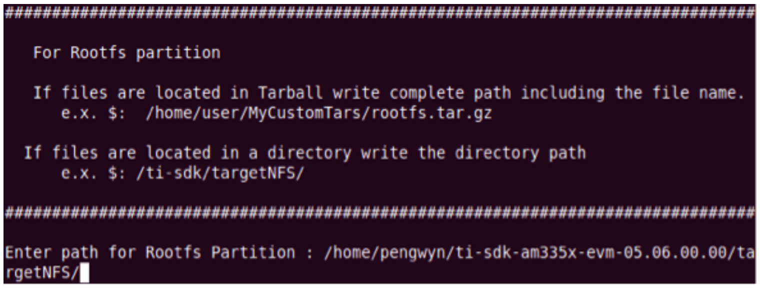
```
Mount the partitions
mount: special device /dev/sdd1 does not exist
mount: special device /dev/sdd2 does not exist

Emptying partitions


Syncing....
################################################################################

        Choose file path to install from

        1 ) Install pre-built images from SDK
        2 ) Enter in custom boot and rootfs file paths

################################################################################
Choose now [1/2] : 2
```

/home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/board-support/built-images

The script will display the files in the folder: MLO, uImage and u-boot.img.

Press y key to continue.

Finally provide filesystem location:



```
################################################################################

   For Rootfs partition

   If files are located in Tarball write complete path including the file name.
      e.x. $:  /home/user/MyCustomTars/rootfs.tar.gz

  If files are located in a directory write the directory path
      e.x. $: /ti-sdk/targetNFS/

################################################################################

Enter path for Rootfs Partition : /home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/ta
rgetNFS/
```

/home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/targetNFS

and confirm it with y key.

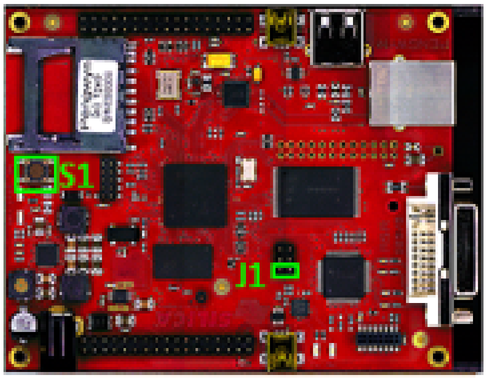
This operation will take minutes.

2.7 Pengwyn Flash Memory

This section will explain how to transfer the data from the sd-card to flash memory of Pengwyn.

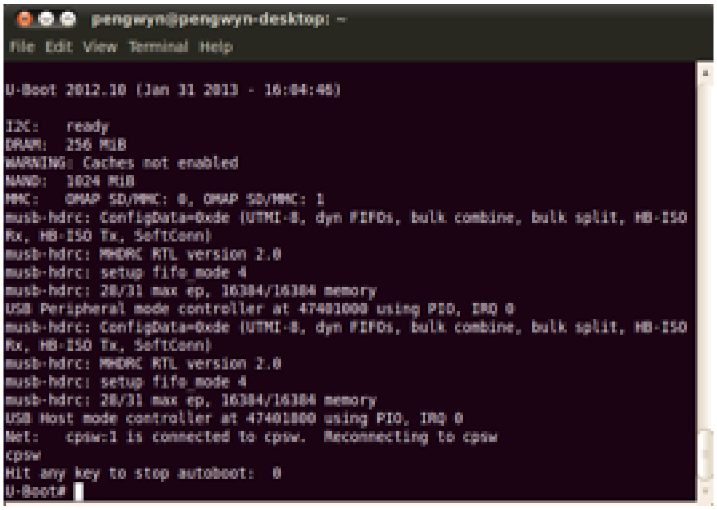
Before going on you must have executed all steps explained in "Connect host PC to Pengwyn board" section. Minicom must work correctly.

1. Remove J1 jumper and insert SD-CARD with prebuilt file



2. Reset with S1 button.

   At the beginning, when you see with Minicom that u-boot procedure starts, press any key to stop it.



```
U-Boot 2012.10 (Jan 31 2013 - 16:04:48)

I2C:   ready
DRAM:  256 MiB
WARNING: Caches not enabled
NAND:  1024 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, bulk combine, bulk split, HB-ISO
Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Peripheral mode controller at 47401000 using PIO, IRQ 0
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, bulk combine, bulk split, HB-ISO
Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Host mode controller at 47401800 using PIO, IRQ 0
Net:   cpsw:1 is connected to cpsw.  Reconnecting to cpsw
cpsw
Hit any key to stop autoboot:  0
U-Boot#
```

3. Erase and upload the FLASH memory with the commands:

```
nand erase.chip
run nandupdate
```

4. Reset board with S1 button. Start Linux operating system, at login enter user name:

   root

   the password is not required.

5. Create flash file system with the automated script:

```
./create_nand_fs.sh
.. image:: /_static/flash3.png
```

6. Shutdown linux with the command:

```
Shutdown -h now
```

7. Remove SD-CARD, insert jumper in J1 and reset the board with S1 button

The system now will restart from NAND flash with new operating system.

2.8 Code Composer Studio V5

Code Composer Studio v5 is currently provided with the Sitara Software Development Kit.

It is based on Eclipse IDE and includes the Remote System Explorer plug-in that enable access the remote target board. This software is already installed and configured in the SDK Pengwyn.

To launch it click the icon “Code Composer Studio v5” on the desktop and ap the Eclipse work environment will appear.

The active project is helloWorld, which we’re going to compile and debug in this tutorial.

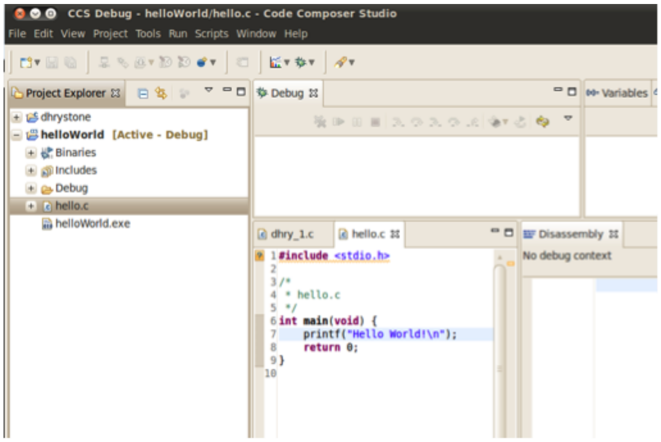
The workspace used is located in:

/home/pengwyn/workspace_v5_1

The project helloWorld is already configured to work with CCS5. If you want create a new project quickly, use this project as template and modify the source code.

From the menu, click View → Project Explorer: the left panel will show the projects available in the workspace.

Open helloWold project and double-click on hello.c file.

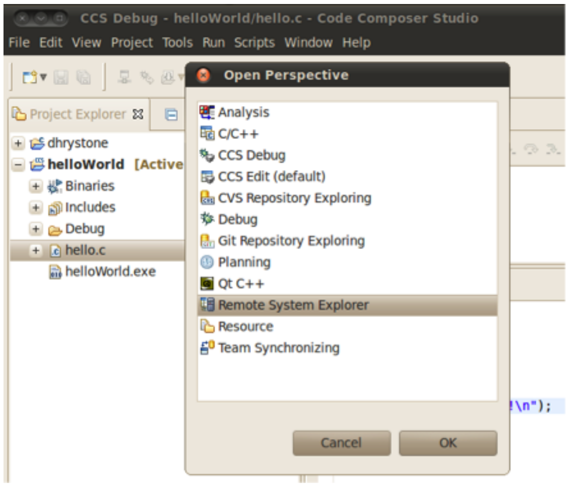

CCS Debug - helloWorld/hello.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Project Explorer
dhrystone
helloWorld [Active - Debug]
Binaries
Includes
Debug
hello.c
helloWorld.exe
Debug
Variables
dhry_1.c
hello.c
Disassembly

```
#include <stdio.h>

/*
 * hello.c
 */
int main(void) {
    printf("Hello World!\n");
    return 0;
}
```

No debug context

To build the project go to Project → Build All.

Go to Window → Open Perspective → Other... → Remote System Explorer to open the Remote System Explorer perspective.



If doesn’t work check Ethernet connection (read “Connect host PC to Pengwyn board” paragraph).

2.8.1 Access Target File System

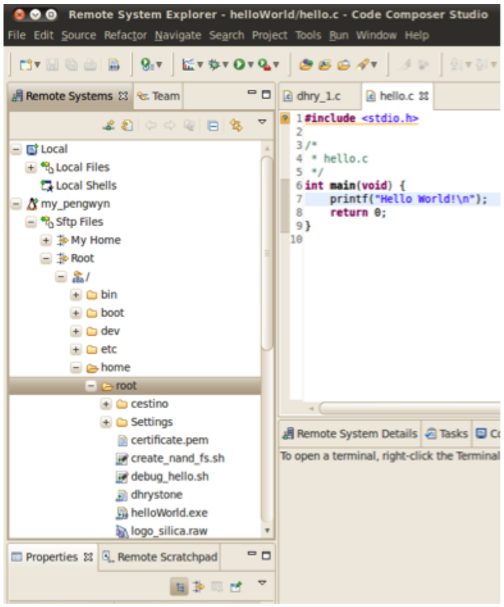
Expand the Root node under Sftp Files node.

The remote system file tree should now show the root directory. You can navigate anywhere in the remote file system down to file level.

Files can be dragged and dropped into the remote file tree. A context menu allows you to create, rename or delete files and folders.

The local file system on the Linux host can also be accessed by expanding the Local node.

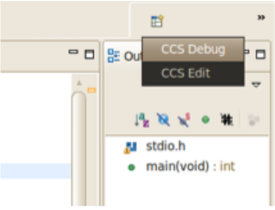
Navigate to the /home/root/ folder on your target board.



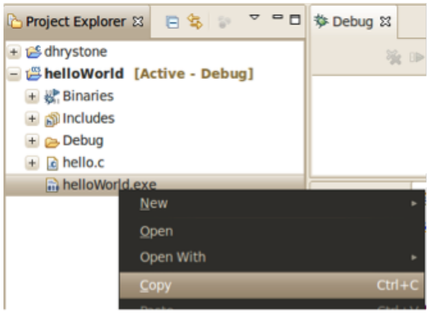
```
#include <stdio.h>

/*
 * hello.c
 */
int main(void) {
    printf("Hello World!\n");
    return 0;
}
```

To open a terminal, right-click the Terminal

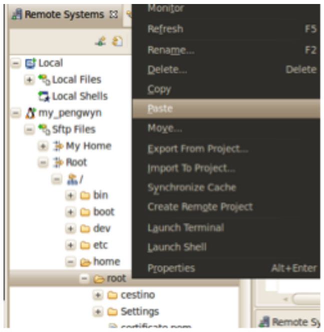
Now we’ll copy our executable to this /home/root target location. First switch back to CCS Edit perspective using the upper right double arrow >>



In Project explorer window, open the Debug group. Select the executable helloWorld.exe, R-click → copy.

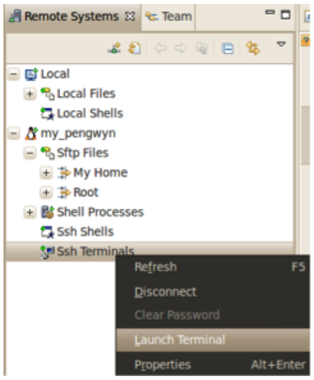


Switch to Remote System Explorer perspective using the upper right double arrow >>. In the Remote Systems window, R-click → paste will copy your executable to the Pengwyn /home/root directory.



2.8.2 SSH Terminal

To open an SSH Terminal view: right click the Ssh Terminals node under the Remote System → my_pengwyn and select Launch Terminal from the context menu.



Type shell commands at the prompt in the terminal window.

Below is a sample command to print the current directory path and list its contents.

In the Console window, Print Working Directory confirm that the binary is there:

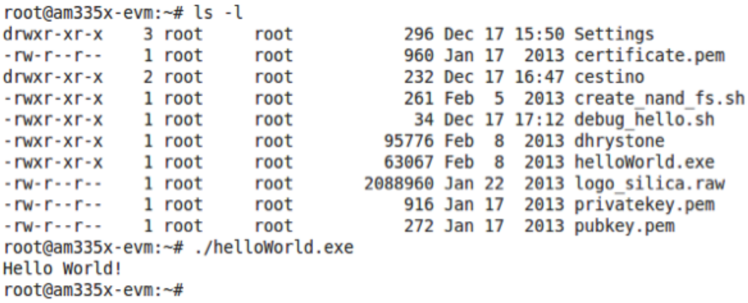
```
ls -l
```

Before you can run your program, you need to make it executable:

```
chmod 755 helloWorld.exe
```

Run your program:

```
./helloWorld.exe
```



```
root@am335x-evm:~# ls -l
drwxr-xr-x    3 root     root           296 Dec 17 15:50 Settings
-rw-r--r--    1 root     root           960 Jan 17  2013 certificate.pem
drwxr-xr-x    2 root     root           232 Dec 17 16:47 cestino
-rwxr-xr-x    1 root     root           261 Feb  5  2013 create_nand_fs.sh
-rwxr-xr-x    1 root     root            34 Dec 17 17:12 debug_hello.sh
-rwxr-xr-x    1 root     root         95776 Feb  8  2013 dhrystone
-rwxr-xr-x    1 root     root         63067 Feb  8  2013 helloWorld.exe
-rw-r--r--    1 root     root       2088960 Jan 22  2013 logo_silica.raw
-rw-r--r--    1 root     root           916 Jan 17  2013 privatekey.pem
-rw-r--r--    1 root     root           272 Jan 17  2013 pubkey.pem
root@am335x-evm:~# ./helloWorld.exe
Hello World!
root@am335x-evm:~#
```

2.8.3 Running the Debug Session

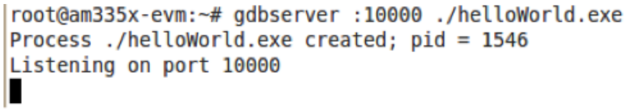
Each time you start the debugger, you must first start the gdbserver program on the target.

Start gdbserver for the helloWorld project with a port number of 10000 (this port number must match the number that was entered in the Debug Configuration).

At the target console command line, type:

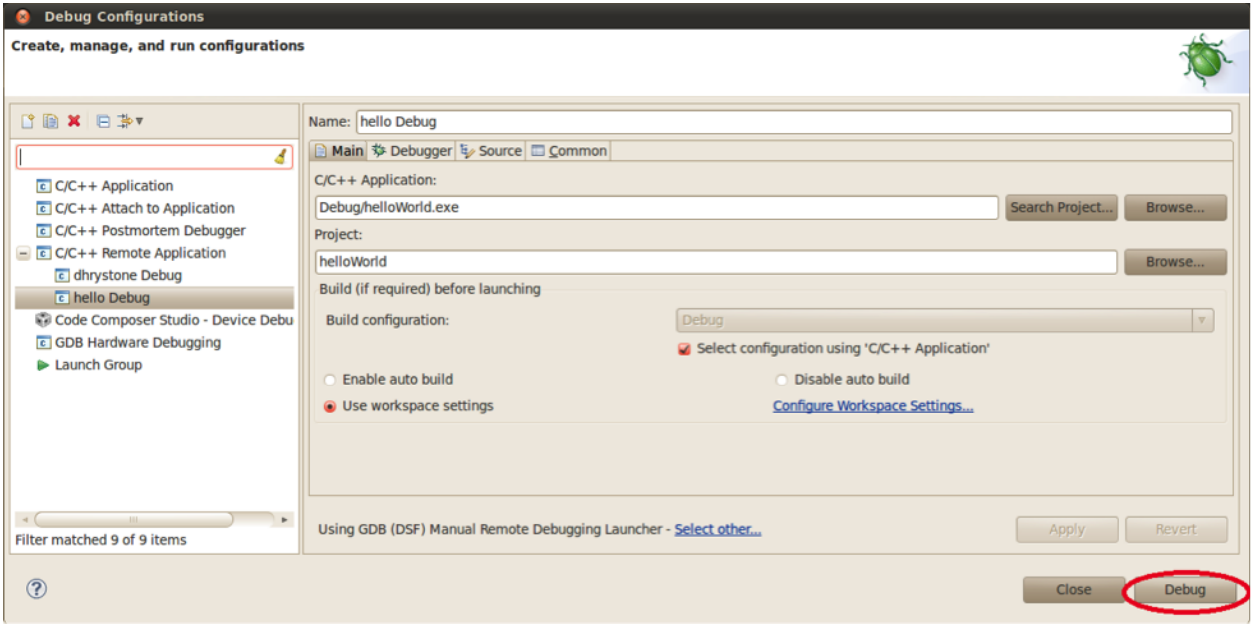
```
gdbserver :10000 ./helloWorld.exe
```

Once started, you should see a response similar to below:



```
root@am335x-evm:~# gdbserver :10000 ./helloWorld.exe
Process ./helloWorld.exe created; pid = 1546
Listening on port 10000
█
```

Go to Run → Debug Configurations, select “hello Debug” and click on Debug button



Debug Configurations

Create, manage, and run configurations

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
  - dhrystone Debug
  - hello Debug
- Code Composer Studio - Device Debu
- GDB Hardware Debugging
- Launch Group

Filter matched 9 of 9 items

Name: hello Debug

Main | Debugger | Source | Common

C/C++ Application: Debug/helloWorld.exe [Search Project...] [Browse...]

Project: helloWorld [Browse...]

Build (if required) before launching

Build configuration: Debug

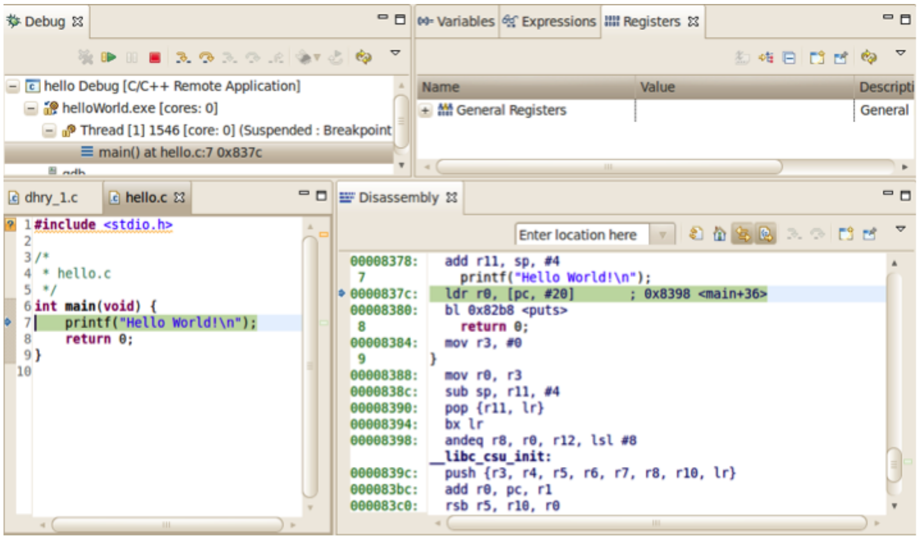
☑ Select configuration using 'C/C++ Application'

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

Using GDB (DSF) Manual Remote Debugging Launcher - Select other...

CCS will change to the CCS Debug perspective. The debug panel will show the running threads and their status. The source code window will show the program halted at the first executable source code line in the main() function.



```
hello Debug [C/C++ Remote Application]
  helloWorld.exe [cores: 0]
    Thread [1] 1546 [core: 0] (Suspended : Breakpoint
      main() at hello.c:7 0x837c
```

```
#include <stdio.h>

/*
 * hello.c
 */
int main(void) {
    printf("Hello World!\n");
    return 0;
}

```

```
00008378:   add r11, sp, #4
 7            printf("Hello World!\n");
0000837c:   ldr r0, [pc, #20]        ; 0x8398 <main+36>
00008380:   bl 0x82b8 <puts>
 8            return 0;
00008384:   mov r3, #0
 9          }
00008388:   mov r0, r3
0000838c:   sub sp, r11, #4
00008390:   pop {r11, lr}
00008394:   bx lr
00008398:   andeq r8, r0, r12, lsl #8
          __libc_csu_init:
0000839c:   push {r3, r4, r5, r6, r7, r8, r10, lr}
000083bc:   add r0, pc, r1
000083c0:   rsb r5, r10, r0
```

The Variables window will show the local variables and their current values.

Like any debug enviroment, you can use breakpoints, use the debugger Step Over and Step Into command icons to step through the source code.

To resume program execution, click the run → resume menu item.

CHAPTER 3

Appendix

3.1 Opkg Basics

Opkg (Open PacKaGe Management) is a lightweight package management system. It is written in C and resembles apt/dpkg in operation. It is intended for use on embedded Linux devices and is used in this capacity in the OpenEmbedded and OpenWrt projects.

- opkg list-installed command to know what packages are installed on the file system. For example use this command with grep:
- opkg list-installed | grep -i name_packet
- opkg search name_packet show where are the files installed of the packet
- opkg whatdepends name_packet show what packets depend on the “name_packet” package
- opkg remove name_packet remove packages, there are important options:
  - -force-depends This option will force the removal of the package but will leave any packages that depend on this package installed
  - -force-removal-of-dependent-packages This option will go up the dependency list and remove all packages in the dependency chain
- opkg install name_packet install the packages

3.2 How to confgure a new project in CCS

Launch Code Composer Studio (aka CCS) using the icon on desktop of your Ubuntu Linux host.

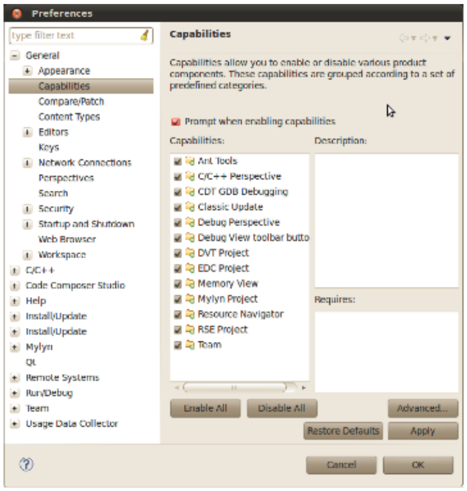
When prompted for the workspace, keep the default one.

3.2.1 Enabling CCS Capabilities

Because we used a new workspace, capabilities need to be enabled to make perspectives selectable in the Window → Open Perspectives list.

After opening CCS with a new workspace:

1. Open the Window → Preferences dialog box
2. Go to the General → Capabilities dialog box.
3. Click Enable All button. The result is shown below.



Capabilities

Capabilities allow you to enable or disable various product components. These capabilities are grouped according to a set of predefined categories.

☑ Prompt when enabling capabilities

Capabilities:

☑ Ant Tools
☑ C/C++ Perspective
☑ CDT GDB Debugging
☑ Classic Update
☑ Debug Perspective
☑ Debug View toolbar butto
☑ DVT Project
☑ EDC Project
☑ Memory View
☑ Mylyn Project
☑ Resource Navigator
☑ RSE Project
☑ Team

This enables all the perspectives in the Window → Open Perspectives → Dialog as shown below.

This is needed to make the C/C++ and Remote System Explorer plug-ins selectable.

Click on “Apply” then “OK”.

3.2.2 Toolchain configuration in CCS

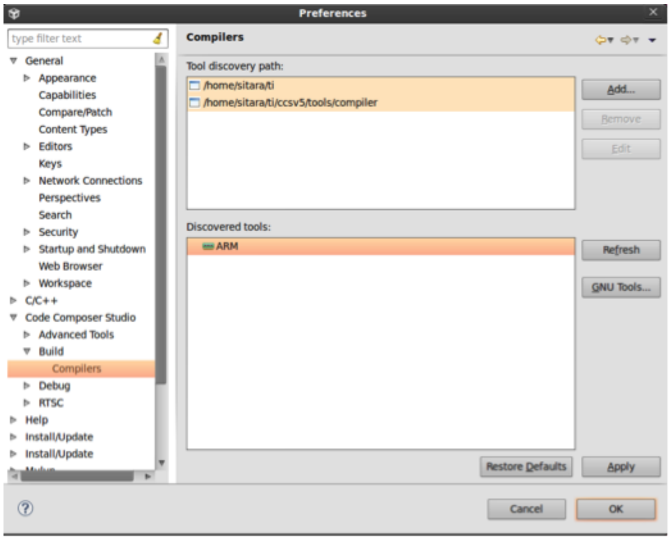
Now let’s create Linux GCC project in CCSv5.

We need first to configure CCS to use the Arm Arago GCC toolchain which has been installed with the SDK, otherwise our project type would use the generic GCC compiler.

All target-related options, typically used for ARM cross-compile (-march=armv7-a, -mtune=cortex-a8 and others), must be specified in the build options before building the project.

We will use the Arago cross-compiler tools instead of the default gcc. The Arago binaries have a prefix that designates the target and the type of output binary format (something like arm-arago-linux-gnueabi-g++).

Go to Window → Preferences → Code Composer Studio → Build → Compilers.



Compilers

Tool discovery path:

/home/sitara/ti
/home/sitara/ti/ccsv5/tools/compiler

Discovered tools:

ARM

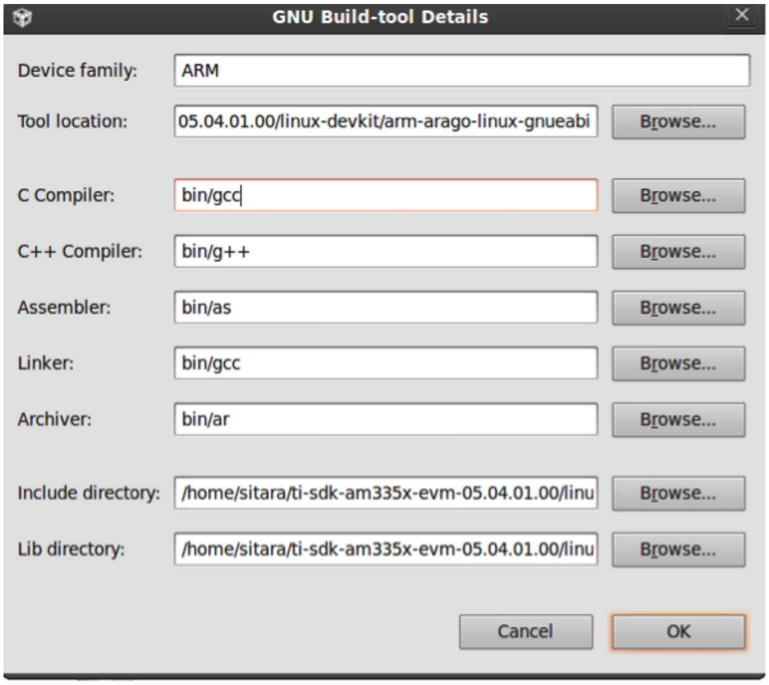
In the Discovered tools section, click on "ARM" then "GNU Tools"

Select Edit

In the Tool installation directory, type the following:

home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/linux-devkit/arm-arago-linux-gnueabi

Then click in any of the below empty boxes; the relative path will get filled in automatically as shown below. Leave the default and click on OK.



Device family: ARM

Tool location: 05.04.01.00/linux-devkit/arm-arago-linux-gnueabi

C Compiler: bin/gcc

C++ Compiler: bin/g++

Assembler: bin/as

Linker: bin/gcc

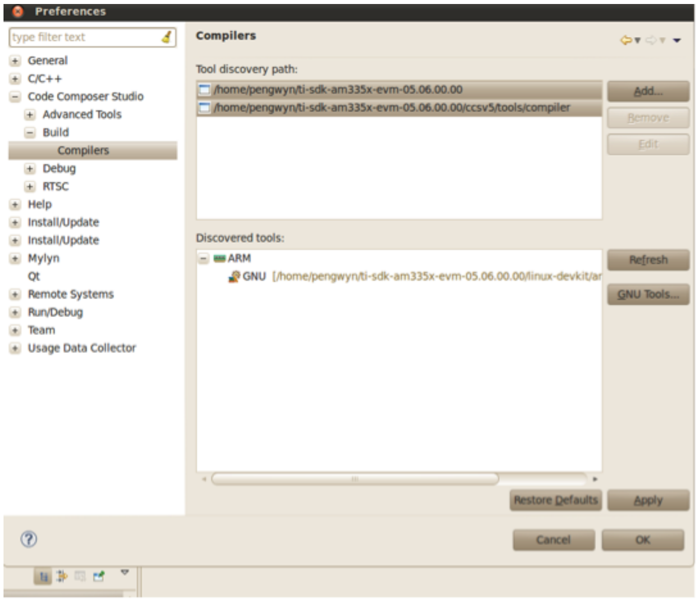
Archiver: bin/ar

Include directory: /home/sitara/ti-sdk-am335x-evm-05.04.01.00/linu

Lib directory: /home/sitara/ti-sdk-am335x-evm-05.04.01.00/linu

Click on OK again to confirm the GNU Build-Tool settings.

Go back to the Preferences window, click on Apply and then OK.



Compilers

Tool discovery path:

/home/pengwyn/ti-sdk-am335x-evm-05.06.00.00

/home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/ccsv5/tools/compiler

Discovered tools:

ARM

GNU [/home/pengwyn/ti-sdk-am335x-evm-05.06.00.00/linux-devkit/ar

3.2.3 Starting your project

Select File → New → Project→Code Composer Studio →CCS project.

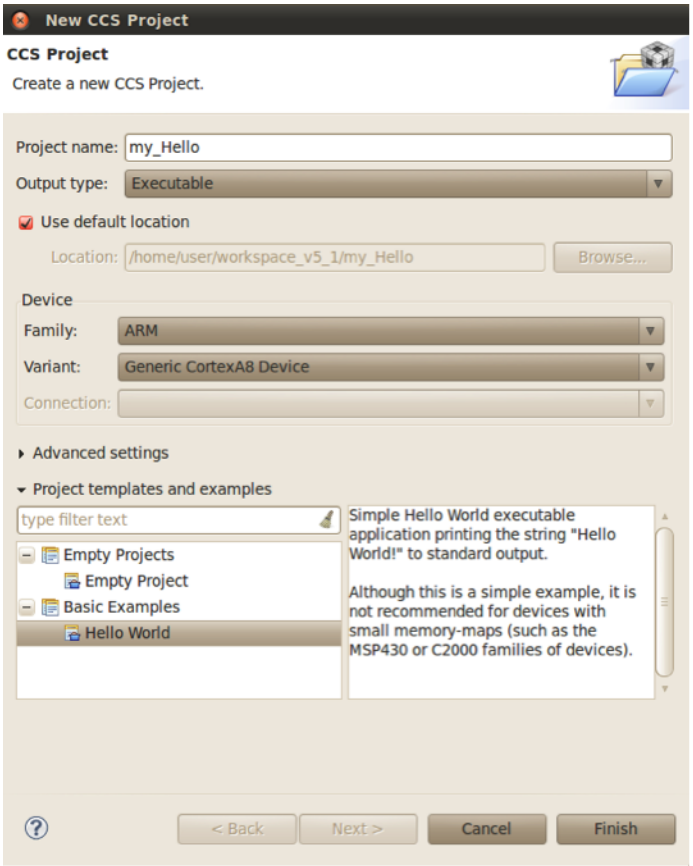
Type in my_Hello as project name.

Output type: Executable.

Change the ARM variant to Generic Cortex-A8 device.

Use the Basic Examples → Hello World project template.

Click Finish.



New CCS Project

CCS Project

Create a new CCS Project.

Project name: my_Hello

Output type: Executable

☑ Use default location

Location: /home/user/workspace_v5_1/my_Hello

Device

Family: ARM

Variant: Generic CortexA8 Device

Connection:

Advanced settings

Project templates and examples

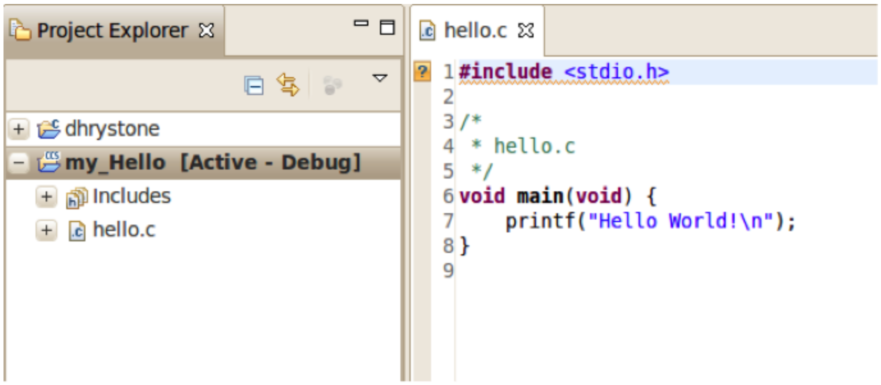
type filter text

- Empty Projects
  - Empty Project
- Basic Examples
  - Hello World

Simple Hello World executable application printing the string "Hello World!" to standard output.

Although this is a simple example, it is not recommended for devices with small memory-maps (such as the MSP430 or C2000 families of devices).

The my_Hello project will be created in your CCS workspace and added to the Project Explorer window (note that it is in bold letters, meaning this is the active project).



Project Explorer

- dhrystone
- **my_Hello [Active - Debug]**
  - Includes
  - hello.c

hello.c

```
#include <stdio.h>

/*
 * hello.c
 */
void main(void) {
    printf("Hello World!\n");
}

```

Note: You can also add source files to the project by R-click on project and selecting “Add Files”.

3.2.4 Build your project

R-click on project name in CCS Project Explorer and select window → Build project.

Check the console view for successful build. . . The executable program my_Hello.exe will be created.

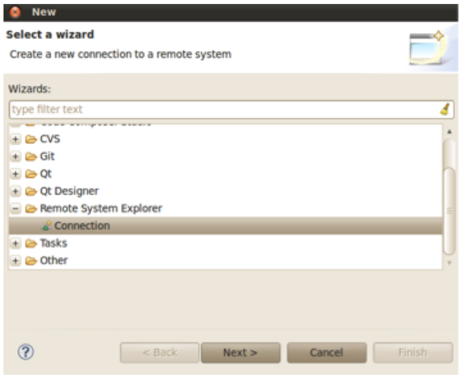
Now you need to copy it to the target filesystem.

Go to Window → Open Perspective → Other... → Remote System Explorer to open the Remote System Explorer perspective.

Now you need to establish a new connection with the Pengwyn.

For that you must run the New Connection Wizard. To open this, click on File → New → Other menu item.

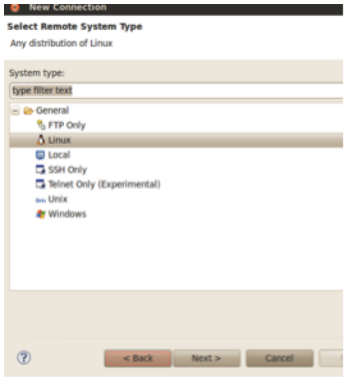
In the dialog below, expand the Remote System Explorer folder, click on Connection and Next.



Select a wizard

Create a new connection to a remote system

Select the Linux system type and click Next.

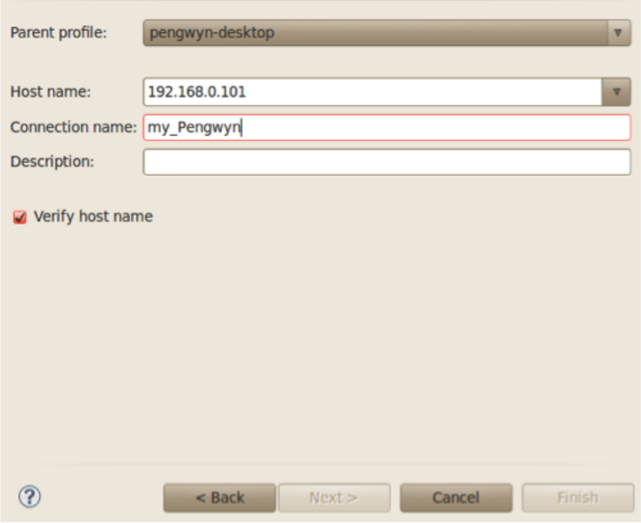


Select Remote System Type

Any distribution of Linux

Next to "Host name" enter the IP address of your Pengwyn: 192.168.0.101

Use "my_pengwyn" as Connection name. Click on Next.



Parent profile: pengwyn-desktop

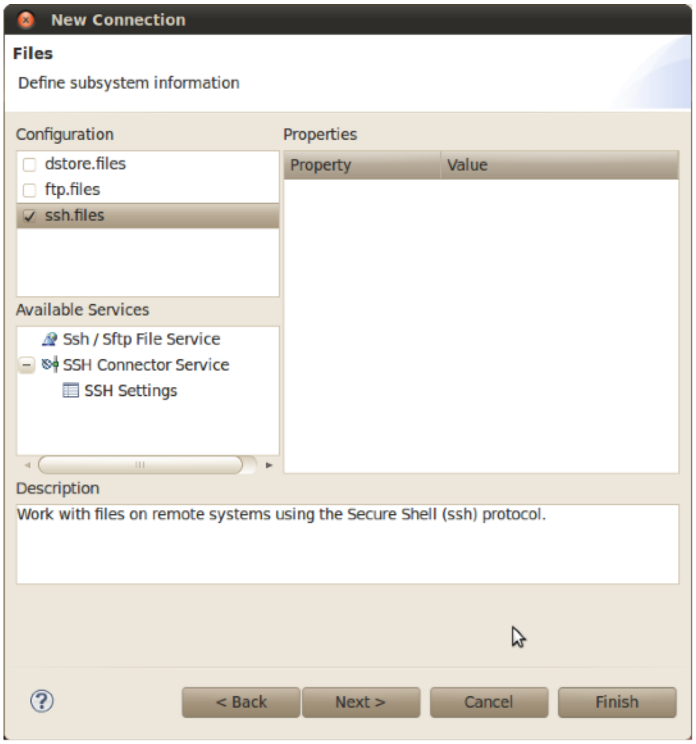
Host name: 192.168.0.101

Connection name: my_Pengwyn

Description:

☑ Verify host name

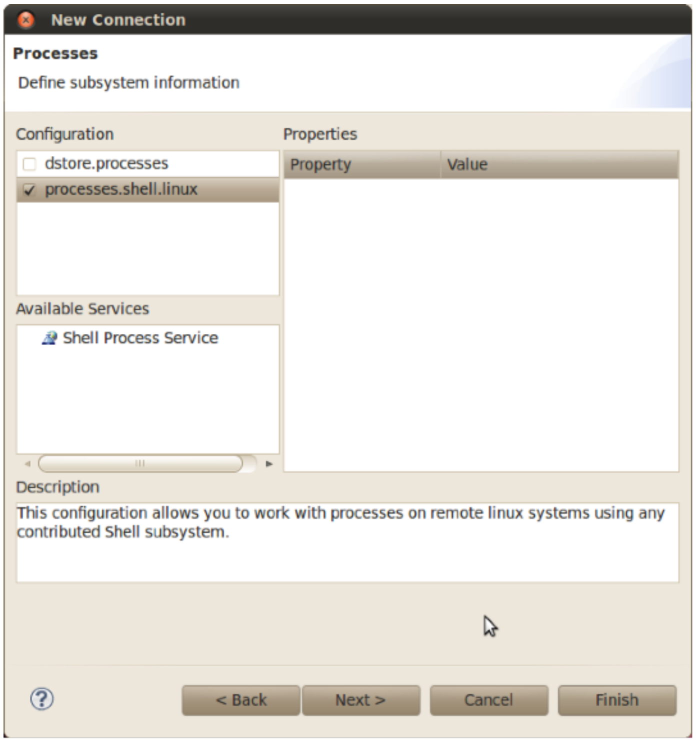
Check ssh.files and click Next.



- □ dstore.files
- □ ftp.files
- ☑ ssh.files

Work with files on remote systems using the Secure Shell (ssh) protocol.

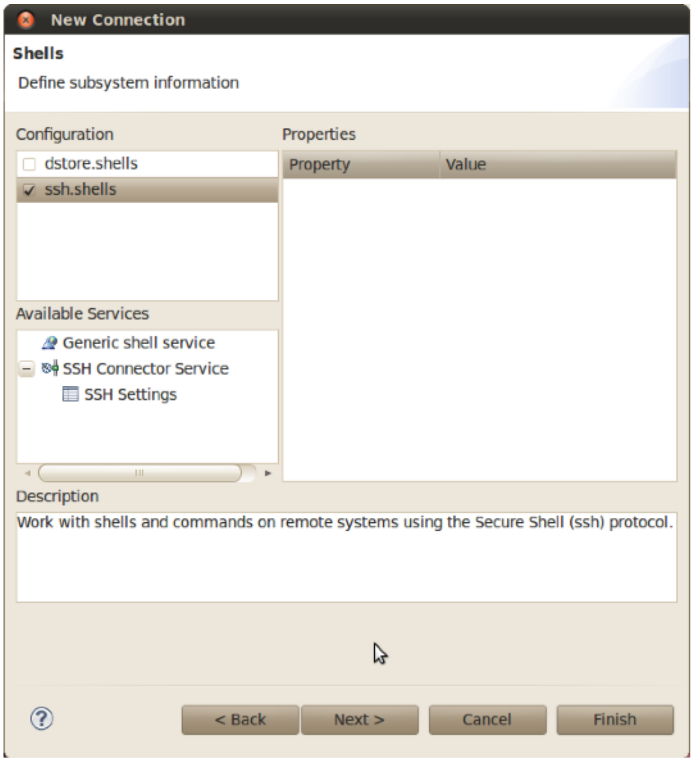
Check processes.shell.linux and click Next.



- □ dstore.processes
- ☑ processes.shell.linux

This configuration allows you to work with processes on remote linux systems using any contributed Shell subsystem.

Check ssh.shells and click Next.


New Connection

Shells

Define subsystem information

Configuration

- □ dstore.shells
- ☑ ssh.shells

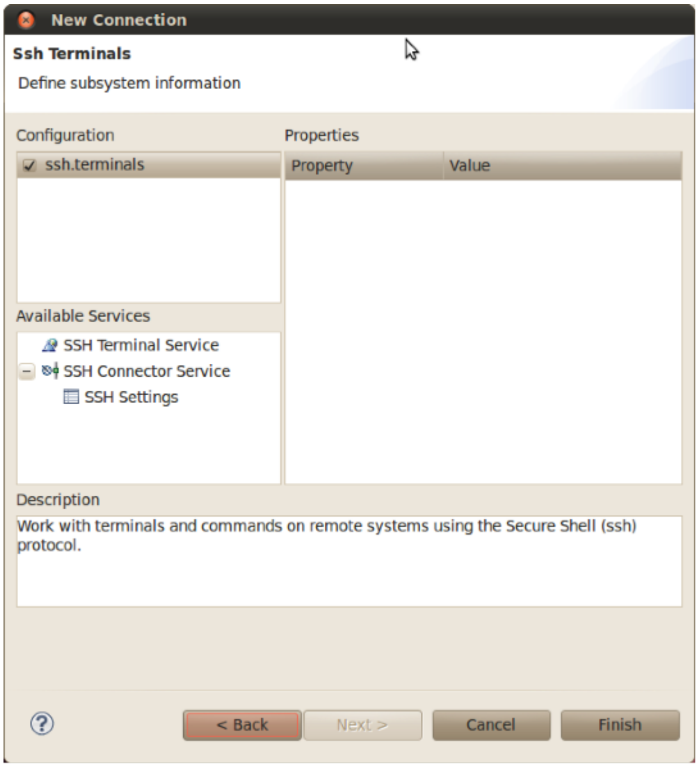
Available Services

- Generic shell service
- SSH Connector Service
  - SSH Settings

Description

Work with shells and commands on remote systems using the Secure Shell (ssh) protocol.

Check ssh.terminals and click Finish to complete the wizard.


New Connection

Ssh Terminals

Define subsystem information

Configuration

- ☑ ssh.terminals

Available Services

- SSH Terminal Service
- SSH Connector Service
  - SSH Settings

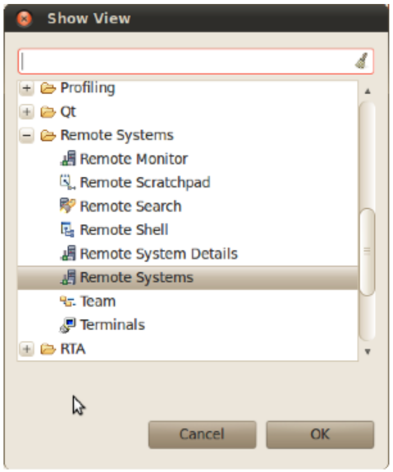
Description

Work with terminals and commands on remote systems using the Secure Shell (ssh) protocol.

3.2.5 Opening the Remote System Explorer View

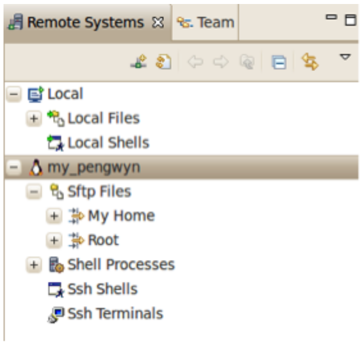
In CCSv5 click the Window → Show View → Other menu item. In the Show View dialog select Remote Systems → Remote Systems then OK.

This adds the Remote Systems view to the current perspective.



A Remote Systems panel appears in the CCS perspective. After a R-click → Detach on the Remote Systems panel and moving it to the left side of the screen the CCS window will look like the screen capture below. The target connection named My_Pengwyn is shown as a tree structure with branches for the various Remote System functions. Communication with the target EVM uses a secure SSH connection.

- Sftp Files - Provides a drag and drop GUI interface to the target file system.
- Shell Processes - Provides a listing of processes running on the remote system and allows processes to be remotely killed.
- Ssh Shells - Provides a Linux shell window for the remote system within CCS.
- Ssh Terminals - Provides a terminal window for the remote system within CCS.

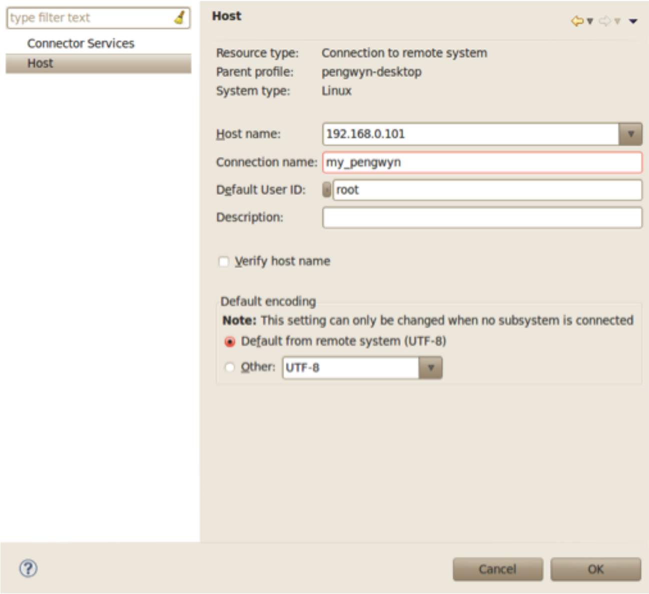


3.2.6 Configuring the Target Pengwyn Connection

After the New Connection Wizard has been completed and the Remote System Explorer view has been opened, the new connection must be configured to communicate with the target EVM.

R-click on the my_pengwyn node and select Properties from the context menu.

After the Properties window opens, click on Host. Change the Default User ID to root and click OK.



Host

Resource type: Connection to remote system
Parent profile: pengwyn-desktop
System type: Linux

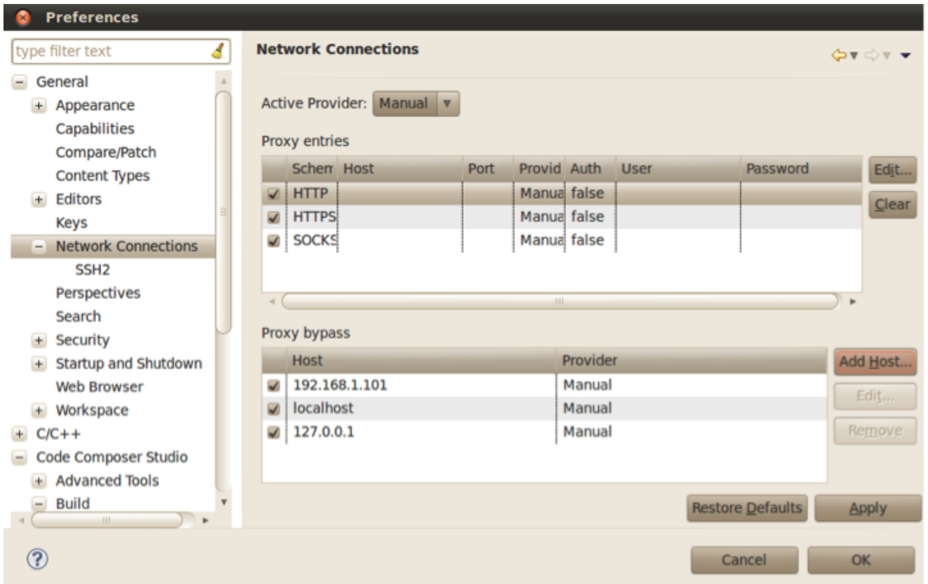
Host name: 192.168.0.101
Connection name: my_pengwyn
Default User ID: root
Description:

☐ Verify host name

Default encoding
Note: This setting can only be changed when no subsystem is connected
● Default from remote system (UTF-8)
○ Other: UTF-8

Click the Window → Preferences menu item. Go to General → Network Connections

In the bottom part of the dialog box, in the Proxy Bypass section, click Add Host..., and add the IP address of target board (192.168.0.101) and click Apply then OK.



Network Connections

Active Provider: Manual

Proxy entries

	Schen	Host	Port	Provid	Auth	User	Password
☑	HTTP			Manua	false		
☑	HTTPS			Manua	false		
☑	SOCKS			Manua	false		

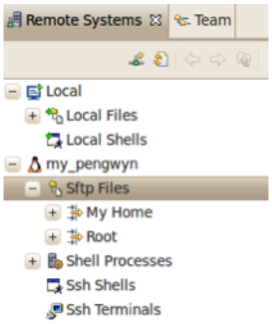
Proxy bypass

	Host	Provider
☑	192.168.1.101	Manual
☑	localhost	Manual
☑	127.0.0.1	Manual

The Remote System Explorer is now ready for use.

The first time the target EVM file system is booted a private key and a public key is created in the target file system. Before connecting to the target EVM the first time, the public key must be exported from the target EVM to the Linux host system.

To open the SSH connection, expand the ROOT node under the Sftp node.



When prompted for the password, click OK (no password).

A warning dialog box will appear: click Yes, and the public key will be exported to the Ubuntu host.

CHAPTER 4

Revision History

Date	Version	Revision
8th February 2013	1.0	Initial Draft Version
11th February 2013	1.1	Release Version
18th February 2013	1.2	Corrections. Added Flash System section.